

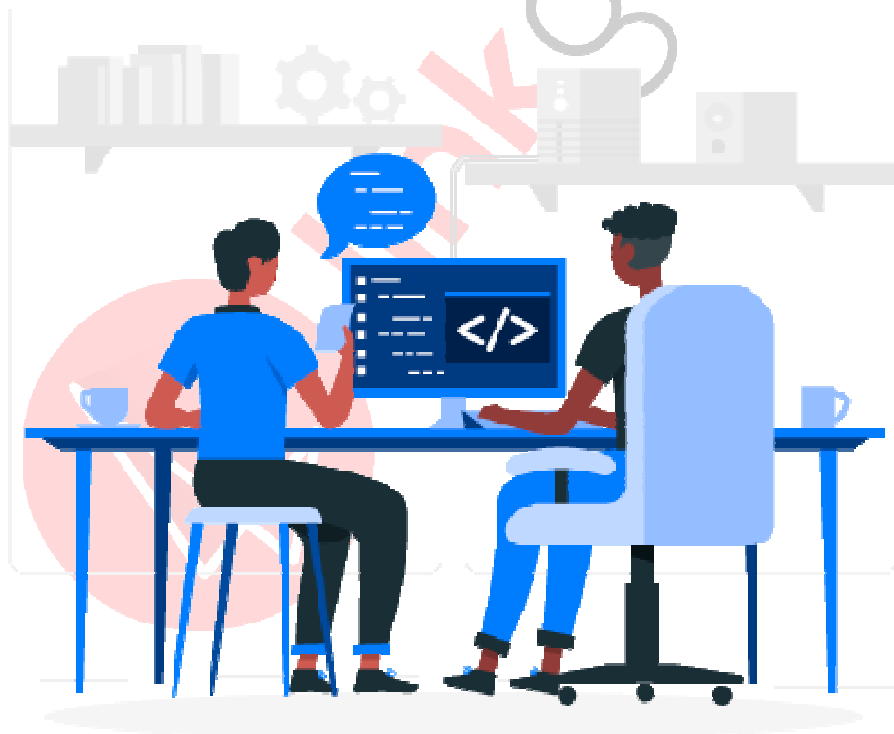
Programiranje u paru

U okviru ove lekcije govorićemo o pojmu programiranja u paru; videćemo kako se ovaj princip rada pojavio, te koje su njegove prednosti i mane. Takođe i koji alati nam danas u poslovnom okruženju omogućavaju realizaciju ovog pristupa.

Ovaj pristup je deo agilnog razvoja softvera i, konkretno, oblasti ekstremnog programiranja, o kojoj smo i ranije govorili. Kao što i sam naziv implicira, programiranje u paru je pristup gde dva programera rade isključivo koristeći jedan računar. U zavisnosti od dogovora, može se koristiti jedan ili više monitora, može svako imati svoj miš i tastaturu, a nekada se u ovom pristupu dele miš i tastatura.

Na ovaj način, jedan programer kreira kod dok drugi služi kao posmatrač i proverava šta je napisano. Dakle, u trenutku pisanja kontroliše ispravnost koda. Posmatrač sagleda veću sliku i razmišlja o tome koji je sledeći korak u kodu. Ove uloge se često zamenjuju, tako da tokom rada svaki od programera ima priliku da piše kod i obrnuto.

Programer koji piše kod se često naziva *driver*, a programer koji nadgleda rad – *navigator*.



Slika 9.1. Ilustracija programiranja u paru¹

¹https://www.freepik.com/free-vector/pair-programming-concept-illustration_8449769.htm#page=1&query=pair%20programming&position=3

Prednosti programiranja u paru

Tokom godina primene ovog principa rada, uočen je veliki broj pozitivnih ishoda. U nastavku navodimo najznačajnije:

- Dve glave su bolje nego jedna – Ukoliko driver (programer koji piše kod) naiđe na problem, uvek postoji još jedan kolega koji je potpuno upoznat sa poslom i koji može da pomogne u rešavanju problema.
- Veća efikasnost – Česta zablude je da ovaj proces usporava realizaciju projekta jer dva programera rade na razvoju jednog programa, jer nam nekako logika govori da je bolje da svaki programer ima svoju celinu koju realizuje. Ali, studije² su pokazale da je u slučaju rada dva programera na jednom programu napredak samo 15% sporiji nego kada rade odvojeno. Pritom stvaraju mnogo pouzdaniji kod sa manje grešaka.
- Manje grešaka u kodu – pojave grešaka prilikom realizacije rada na ovaj način su daleko ređe. U radu po ovom principu, driver ostaje fokusiran na kod koji piše, dok navigator rešava ostale poslove, komunikaciju sa drugim kolegama u timu i generalno sprečava prekide u radu koje bi driver u suprotnom imao. Pored ovoga se greške i u većoj meri uočavaju kada se kod kreira, u odnosu na to da bi možda prvi put bile uočene u QA fazi softvera.
- Efikasan način prenosa i deljenja znanja – Parovi programera tokom celog procesa razvoja uče jedno od drugog. Manje iskusni programeri upareni sa seniorima u kompaniji dobijaju instrukcije licem u lice, što je značajno bolji način učenja od tutorijala i drugih resursa na internetu. Takođe, bez obzira na iskustvo programera, dele se primeri dobre prakse i bolje tehnike realizacije funkcija.
- Razvoj interpersonalnih veština – Primenom ovog principa rada, članovi tima mnogo više komuniciraju i uče se vrednostima timskog rada, što uvećava kretanje informacija između članova tima i podiže dinamiku radnog okruženja.
- Više ljudi poznaje detalje programa – U slučaju prekida rada jednog programera, drugi programer može preuzeti njegov posao, gde bi u suprotnom rad na programu ili nekom modulu bio zaustavljen dok se drugi programer ne upozna sa već napisanom logikom.

Izazovi u primeni programiranja u paru

Naravno, kako imamo prednosti, uvek postoje i nedostaci ili prepreke koje je potrebno zaobići prilikom realizacije rada na ovaj način:

- Najčešći problem je postizanje maksimalnog angažovanja od strane oba programera tokom celog trajanja zadatka izrade. Ukoliko jedan od programera ne doprinosi dovoljno, benefiti korišćenja programiranja rada u paru jednostavno nestaju.
- Programiranje u paru često iziskuje „programiranje naglas“, gde driver govori šta planira da uradi ili šta trenutno kuca. Ukoliko programer nema naviku da ovako nešto radi i ne želi da je stekne, onda programiranje u paru postaje teže za realizaciju.
- Programiranje u paru nije nešto što može biti nametnuto timu. Kako je ovaj princip rada visoko interaktivan i društven, pojedinci se jednostavno ne mogu uvek povezati na ovaj način sa kolegom iz tima. Ovo, naravno, može biti do same ličnosti ili postojanja prethodnih konflikata.

² <https://www.sciencedirect.com/science/article/abs/pii/S0950584909000123>

Do sada je praksa programiranja u paru bila da na radnom mestu i za jednim stolom sede dva programera. Tokom prethodnih godina, u većoj meri se za pozicije programera uveo rad na daljinu, što je uvećalo potražnju za softverom koji omogućava isti vid interakcije i kolaboracije putem interneta. Samim tim smo dobili i novi pojam – *remote pair programming*, tj. programiranje u paru na daljinu.

Pitanje

Programer koji nadgleda pisanje koda naziva se driver.

- Tačno.
- **Netačno.**

Objašnjenje:

Programer koji piše kod se često naziva driver, a programer koji nadgleda rad navigator. Navigator služi kao posmatrač i proverava šta je napisano, tj. u trenutku pisanja kontroliše ispravnost koda. Navigator sagledava veću sliku i razmišlja o tome koji je sledeći korak u kodu.

Remote pair programming

Do ove pojave se došlo veoma jednostavno. Ako cela kompanija radi od kuće i klasično programiranje u paru nije moguće, kako zaobići ovu prepreku? Kao što ste mogli da vidite, za kompanije je ovo vrlo vredan pristup, a u poslednjim godinama postao je još vredniji, pa ako radnici ne mogu komunicirati licem u lice, zašto ne bi iskoristili već postojeće tehnologije da nastave praksu koju su imali u organizaciji.

Suštinski, programiranje u paru na daljinu ima iste benefite kao i klasično programiranje u paru. Dakle: deljenje znanja, bolji kod, lakše otkrivanje grešaka. Prema trenutnom broju alata i njihovim brojkama preuzimanja, ovaj pristup će se održati i u narednim godinama.

Alati koji se koriste za programiranje u paru

Kao što smo rekli, u klasičnom pristupu naročitih alata nema – koristi se isti računar, a periferije računara se mogu menjati u zavisnosti od toga šta odgovara kojem paru. Sa radom na daljinu, logika je malo drugačija. Alata za kolaboraciju postoji na hiljade, od opštih alata za komunikaciju, poput Skypea, preko specijalizovanih alata kreiranih za potrebe rada u paru.

Stoga se došlo do klasifikacije po nivoima, gde svaki naredni nivo predstavlja veće odstupanje od klasičnog driver/navigator pristupa, ali ujedno svaki nivo donosi bolju saradnju i veću produktivnost.

- Alati nultog nivoa – Zoom, Google Hangouts, Skype
 - Na ovaj nivo možemo smestiti sve alate za video-konferencije koji imaju mogućnost prikazivanja ekrana. Programeri ovde sarađuju tako što driver deli svoj ekran, a navigator prati realizaciju. U ovom slučaju, audio-video komunikacija je stalna.
- Alati prvog nivoa – Remote Desktop Services, AnyDesk, Teamviewer
 - Alati za video-konferencije koji omogućavaju kontrolu računara učesnika.

- Alati drugog nivoa – plugini koji omogućavaju rad u istom fajlu kroz tekst editore i IDE
 - Alati ovoga nivoa pružaju najbolje mogućnosti saradnje u samom kodu. Vrlo lako se menjaju uloge i svako od učesnika može da pristupa terminalima, izvršavanju koda i slično. Postoji mogućnost da oba programera unose kod u isto vreme na različitim delovima fajla.

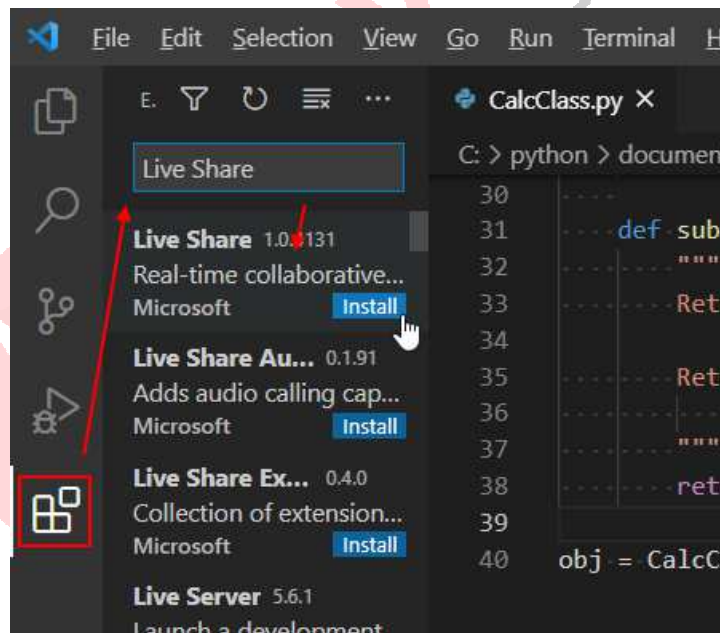
Trenutno većina popularnih tekst editora i razvojnih okruženja podržava neki vid programiranja u paru, recimo:

- <https://visualstudio.microsoft.com/services/live-share/> – Visual Studio i Visual Studio Code;
- <https://www.codetogether.com> – VS Code, Eclipse;
- <https://floobits.com> – IntelliJ, Atom, Sublime Text.

Integracija podrške za programiranje u paru – Visual Studio Code

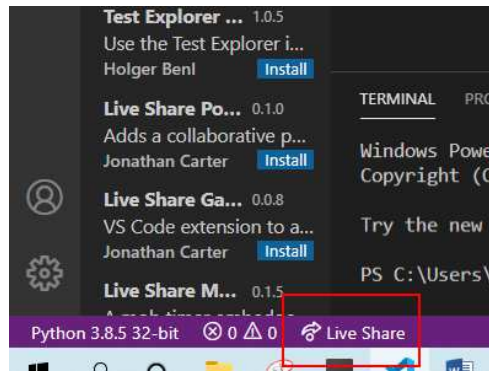
Kao što ste mogli da vidite, Visual Studio Code ima podršku za programiranje u paru, i to u vidu ekstenzije Live Share. Stoga ćemo u narednom delu lekcije i pokazati kako možemo izvršiti ovo podešavanje.

Krećemo od samog editora i njegove sekcije Extensions. U okviru ove sekcije unosimo naziv plugina/ekstenzije – Live Share:



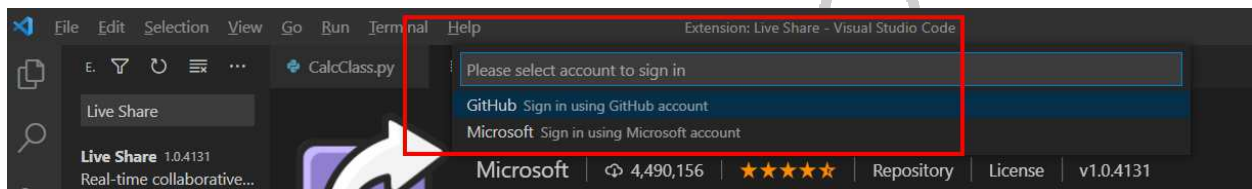
Slika 9.2 – Izbor plugina Live Share

Na listi pronalazimo ovaj plugin i zatim biramo *Install*. Nakon uspešne instalacije, u donjem levom uglu programa možemo videti novu opciju – *Live Share* (slika 9.3).



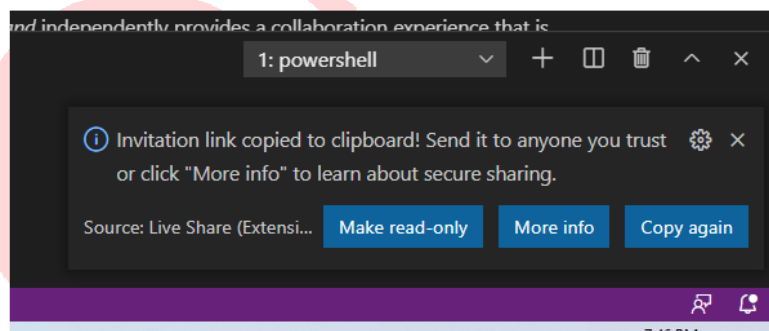
Slika 9.3. Lokacija prečice za pokretanje plugina

Izborom opcije nudi nam se dijalog prozor za login. Da biste ostvarili login, potreban je ili GitHub nalog ili Microsoft nalog.



Slika 9.4. Izbor naloga za login

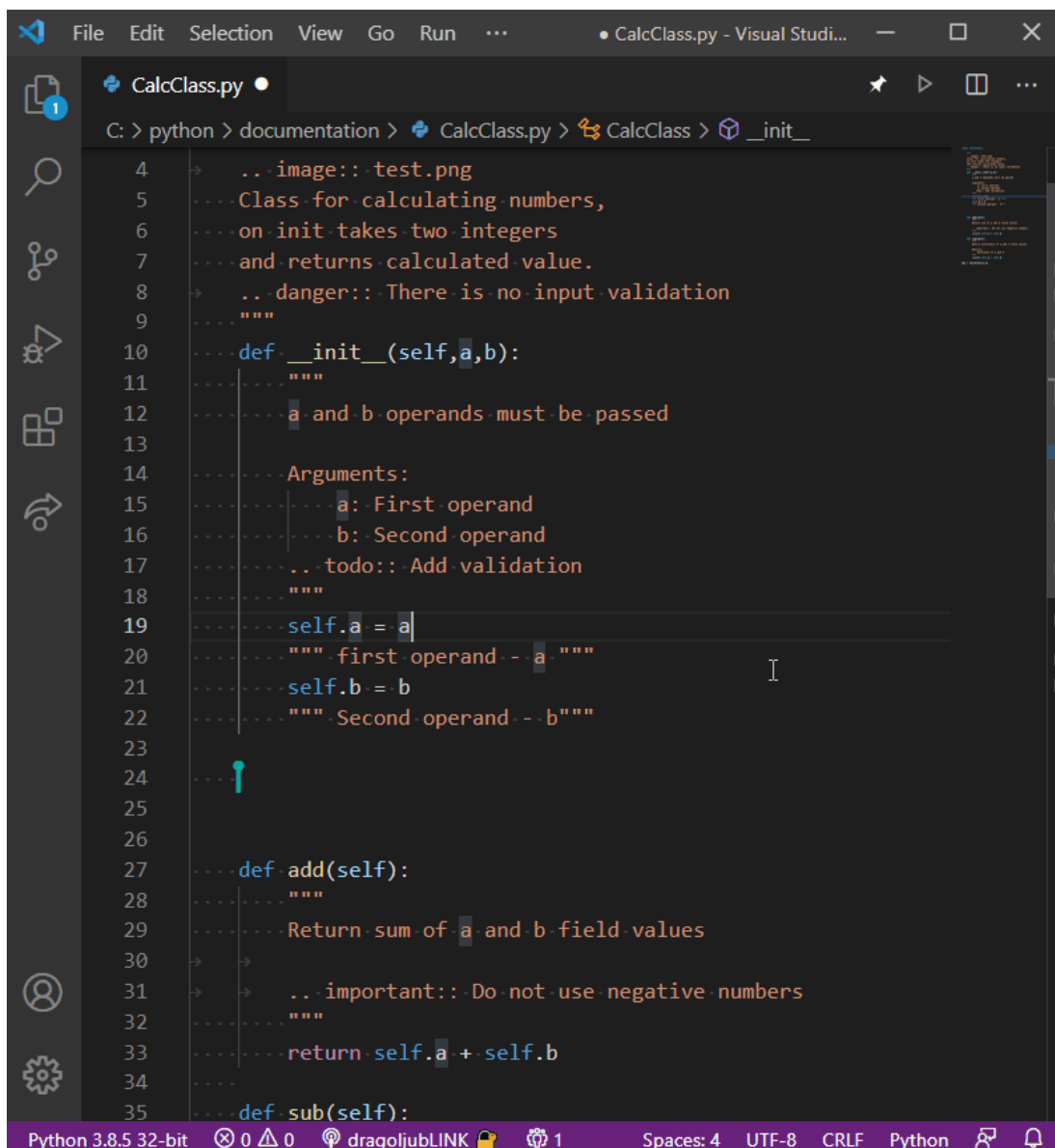
Izborom jedne od dve opcije automatski se otvara stranica za login u okviru pregledača. Nakon uspešnog logina na jedan od ove dve vrste naloga, u donjem desnom uglu editora pojavljuje se sledeći prozor:



Slika 9.5. Obaveštenje o generisanju linka za partnera

U ovom prozoru dobijamo obaveštenje da je generisan link za poziv na saradnju. Link je kopiran u clipboard i sada je samo potrebno proslediti link saradniku. Saradnik na svom računaru mora imati instaliran Visual Studio Code i to je dovoljno. Kako pristupi linku, otvara se Visual Studio Code i započinje instalacija plugina. Nakon uspešne instalacije, saradnik unosi samo username koji će se prikazivati dok sarađujete.

U okviru sledeće animacije možete videti kratak prikaz kako izgleda saradnja na kodu unutar Visual Studio Codea:



```
File Edit Selection View Go Run ... CalcClass.py - Visual Studi...
CalcClass.py
C: > python > documentation > CalcClass.py > CalcClass > __init__
4  →  ..image:: test.png
5  ... Class for calculating numbers,
6  ... on init takes two integers
7  ... and returns calculated value.
8  →  ..danger:: There is no input validation
9  ...
10 ... def __init__(self,a,b):
11 ...     """
12 ...     a and b operands must be passed
13 ...
14 ...     Arguments:
15 ...     a: First operand
16 ...     b: Second operand
17 ...     ..todo:: Add validation
18 ...     """
19 ...     self.a = a
20 ...     """first operand - a """
21 ...     self.b = b
22 ...     """Second operand - b"""
23 ...
24 ...
25 ...
26 ...
27 ... def add(self):
28 ...     """
29 ...     Return sum of a and b field values
30 ...
31 ...     ..important:: Do not use negative numbers
32 ...     """
33 ...     return self.a + self.b
34 ...
35 ... def sub(self):
```

Python 3.8.5 32-bit 0 0 dragoljubLINK 1 Spaces: 4 UTF-8 CRLF Python

Animacija 9.1. Prikaz kolaboracije unutar Visual Studio Codea

Dakle, u okviru Visual Studio Codea može se vršiti istovremeno pisanje koda. Pored ovoga, plugin omogućava audio-komunikaciju. Na ovaj način postiže se vrlo efikasno programiranje u paru. Princip i dalje ostaje isti: osoba koja deli link je driver – dakle, fajl je kod njega i on čuva napredak na njemu, dok je osoba koja ulazi u saradnju putem linka. Navigator može da pravi izmene i učestvuje u komunikaciji.

Rezime

- Programiranje u paru je deo agilnog razvoja softvera i, konkretno, oblasti ekstremnog programiranja. Kao što i sam naziv implicira, programiranje u paru je pristup gde dva programera rade isključivo koristeći jedan računar.
- U okviru rada po ovom principu, jedan programer kreira kod, dok drugi služi kao posmatrač i proverava šta je napisano. Programer koji piše kod se često naziva driver, a programer koji nadgleda rad navigator.
- Programiranje u paru donosi mnogo prednostu u radu; pre svega, veća je efikasnost programera; grešaka je manje; dolazi do deljenja znanja i razvoja interpersonalnih veština.
- Remote pair programming predstavlja programiranje u paru na daljinu, gde postoje isti benefiti kao kod klasičnog programiranja u paru, a osnova razlika je u tome što se umesto korišćenja zajedničkog računara koriste alati za kolaboraciju.
- Sa velikim brojem alata za kolaboraciju, vremenom se došlo do klasifikacije po nivoima, gde svaki naredni nivo predstavlja veće odstupanje od klasičnog driver/navigator pristupa, ali ujedno svaki nivo donosi bolju saradnju i veću produktivnost.

